Neural Network & Deep Learning ICP 3

Student name: SRINIVAS MUSINURI

Student Id: 700758813

Git Hub Link:

https://github.com/srinivasmusinuri/700758813_NNDL_ICP3

Video Link:

https://drive.google.com/file/d/1uqMMjGkqvyhMYyI_OJ8Kg0uwOae50sK4/view?usp=drive_link

1. Create a class Employee and then do the following
   • Create a data member to count the number of Employees
   • Create a constructor to initialize name, family, salary, department
   • Create a function to average salary
   • Create a Fulltime Employee class and it should inherit the properties of Employee class
   • Create the instances of Fulltime Employee class and Employee class and call their member functions

```
class Employee:
    employees_count = 0

    def __init__(self, name, family, salary, department):
        self.name = name
        self.family = family
        self.salary = salary
        self.department = department
        Employee.employees_count += 1
```

```
  @classmethod
  def avg_salary(cls, employees):
      total_sal = sum(employee.salary for employee in employees)
      if len(employees) > 0:
        return total_sal / len(employees)
      else:
        return 0
```

```python
# • Create a Fulltime Employee class
class FulltimeEmployee(Employee):
    def __init__(self, name, family, salary, department):
        super().__init__(name, family, salary, department)
```

```python
# Creating the instances for Employee class
employee1 = Employee("srinivas", "musinuri", 50000, "HR")
employee2 = Employee("raj", "singh", 60000, "Finance")
employee3 = Employee("zakir", "khan", 55000, "IT")

# Creating the instances for FulltimeEmployee class
fulltime_employee1 = FulltimeEmployee("Rakesh", "prana", 70000, "Marketing")
fulltime_employee2 = FulltimeEmployee("iqbal", "hussain", 75000, "Sales")
```

Input & Output:

```python
# Calling the respective member functions
employees = [employee1, employee2, employee3, fulltime_employee1, fulltime_employee2]
avg_salary = Employee.avg_salary(employees)

print(f"Employees Count: {Employee.employees_count}")
print(f"Average salary: ${avg_salary:.2f}")
```

```
Employees Count: 5
Average salary: $62000.00
```

2. Numpy

   Using NumPy create random vector of size 20 having only float in the range 1-20. Then reshape the array to 4 by 5

   Then replace the max in each row by 0 (axis=1)

   (You can NOT implement it via for loop)

```python
import numpy as np

# Creating a random vector here
random_vector = np.random.uniform(1, 20, 20)
print(random_vector)
```

```
[19.15961656 13.60842926 12.95407168  6.70125638  9.58354983  3.5971625
  1.17503721 16.2196708  19.70490435  3.22582169 15.25245828  5.00318652
  5.92580498 19.91145214  8.17013975 16.40998817 13.71456156  1.20455075
 12.38913539 17.16994938]
```

```python
# Reshaping the generated array to 4 by 5
reshape_array = random_vector.reshape(4, 5)
print(reshape_array)
```

```
[[19.15961656 13.60842926 12.95407168  6.70125638  9.58354983]
 [ 3.5971625   1.17503721 16.2196708  19.70490435  3.22582169]
 [15.25245828  5.00318652  5.92580498 19.91145214  8.17013975]
 [16.40998817 13.71456156  1.20455075 12.38913539 17.16994938]]
```

```python
# Insteading of looping, we are getting the indices of the maximum values in each row of array

indices_position = np.argmax(reshape_array, axis=1)
print(indices_position)
```

```
[0 3 3 4]
```

```python
# Replace the maximum values with 0
# Using np.arrange function, we are generating an array of values from 0 to 3
# Using advanced indexing, here we are combing the postion indices and our newly generated indices and assigning 0:

reshape_array[np.arange(4), indices_position] = 0

print(reshape_array)
```

```
[[ 0.         13.60842926 12.95407168  6.70125638  9.58354983]
 [ 3.5971625   1.17503721 16.2196708   0.          3.22582169]
 [15.25245828  5.00318652  5.92580498  0.          8.17013975]
 [16.40998817 13.71456156  1.20455075 12.38913539  0.        ]]
```

Output:

```
[[ 0.         13.60842926 12.95407168  6.70125638  9.58354983]
 [ 3.5971625   1.17503721 16.2196708   0.          3.22582169]
 [15.25245828  5.00318652  5.92580498  0.          8.17013975]
 [16.40998817 13.71456156  1.20455075 12.38913539  0.        ]]
```